Variable-Length Particle Swarm Optimization for Feature Selection on High-Dimensional Classification

Binh Tran, *Member, IEEE*, Bing Xue, *Member, IEEE*, and Mengjie Zhang, *Senior Member, IEEE*

***Abstract*—With a global search mechanism, particle swarm optimization (PSO) has shown promise in feature selection (FS). However, most of the current PSO-based FS methods use a fix-length representation, which is inflexible and limits the performance of PSO for FS. When applying these methods to high-dimensional data, it not only consumes a significant amount of memory but also requires a high computational cost. Overcoming this limitation enables PSO to work on data with much higher dimensionality which has become more and more popular with the advance of data collection technologies. In this paper, we propose the first variable-length PSO representation for FS, enabling particles to have different and shorter lengths, which defines smaller search space and therefore, improves the performance of PSO. By rearranging features in a descending order of their relevance, we facilitate particles with shorter lengths to achieve better classification performance. Furthermore, using the proposed length changing mechanism, PSO can jump out of local optima, further narrow the search space and focus its search on smaller and more fruitful area. These strategies enable PSO to reach better solutions in a shorter time. Results on ten high-dimensional datasets with varying difficulties show that the proposed variable-length PSO can achieve much smaller feature subsets with significantly higher classification performance in much shorter time than the fixed-length PSO methods. The proposed method also outperformed the compared non-PSO FS methods in most cases.**

***Index Terms*—Classification, data mining, feature selection (FS), high-dimensional data, particle swarm optimization (PSO).**

I. Introduction

RECENTLY, feature selection (FS) has become an essential technique in data preprocessing especially on high-dimensional data. With the tremendous growth in data collection technologies, the number of features collected in many machine learning applications becomes increasingly larger. However, the existence of irrelevant and redundant features in these datasets may obscure the relevant ones, which significantly degrades the performance of many learning algorithms. Therefore, with the aim of eliminating irrelevant and redundant features, FS helps in improving the accuracy and interpretability of the learned models, shortening the learning time, and reducing the storage space of the dataset [1].

Researchers have proposed a large number of FS methods for classification problems, which can be classified into wrapper and filter approaches [2]. While a wrapper method evaluates the goodness of a feature subset using a classification algorithm, a filter method is based solely on the intrinsic characteristic of the training data. Therefore, wrappers can usually obtain better classification performance than filters, but with higher computation time. Filter methods are also said to be more general than wrappers. Therefore, a combination of these two approaches has also been proposed to combine their strengths [3].

Although being studied for decades, FS is still a challenging task especially on high-dimensional data due to its huge search space. FS is a combinatorial optimization problem with 2^n possible combinations, where n is the number of original features. In other words, the search space grows exponentially with the number of features.

By ranking features individually, feature ranking or feature weighting methods [4] usually scale well with high-dimensional data. Features are ranked based on their degrees of relevance to the target concept. Then a predefined number of top-ranked features will be selected to form the final subset. However, it is difficult to determine an appropriate number of features to select without a certain amount of domain knowledge or extensive trials. Furthermore, there can be two-way, three-way, or multiway complex interactions among features [5]. An individually weakly relevant feature may become highly useful when combined with other features and vice versa. By evaluating features independently, these methods cannot handle feature interactions. In addition, the top-ranked features may be redundant, which may degrade the performance of classification algorithms. An improvement of feature ranking approaches is to append a second stage, where a heuristic search is applied to the top-ranked features to remove less relevant and redundant features [6]. However, since features are individually ranked in the first stage, this approach may fail to identify multiway feature interactions.

In contrast with feature ranking, feature subset selection methods can evaluate the whole feature subset at once, which

Manuscript received March 21, 2018; revised June 28, 2018; accepted August 22, 2018. Date of publication September 10, 2018; date of current version May 29, 2019. This work was supported in part by the Marsden Fund of New Zealand Government under Contract VUW1509 and Contract VUW1615, in part by the Huawei Industry Fund under Grant E2880/3663, and in part by the University Research Fund at Victoria University of Wellington under Grant 209862/3580 and Grant 213150/3662. *(Corresponding author: Mengjie Zhang.)*

The authors are with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: binh.tran@ecs.vuw.ac.nz; bing.xue@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2018.2869405

can better deal with feature interactions. Sequential forward selection (SFS) [7] and backward FS methods (SBS) [8] are typical feature subset selection methods. SFS (SBS) gradually adds (removes) features until no further improvement. While SFS can be efficient in high-dimensional data, SBS is too expensive to apply to these datasets [3]. However, using a greedy search, SFS and similar methods feature subsets are prone to be stuck in local optima, especially in a search space with thousands of features. A global search technique is needed to explore this huge search space better.

Particle swarm optimization (PSO) proposed by Eberhart and Kennedy [9] is a population-based algorithm which is well-known with global search ability. Simulating the social behavior of bird flocking, PSO works by maintaining a swarm of particles, each of which represents a candidate solution. By communicating their best found solutions, these particles can fly toward more fruitful areas and discover better solutions. PSO has been applied and shown promise in many problems [10], [11]. However, most of its applications are usually on low dimensionality with tens or hundreds of features [12], [13]. Its performance on high-dimensional data with thousands of features or more is still limited due to the following limitations which motivate us to propose a new PSO-based FS method.

A. Motivations

First of all, most of the PSO-based FS methods in the literature use the fix-length representation. In other words, all the particles in the population have the length which is equal to the original number of features. With this representation, PSO usually requires a significant amount of memory and computation time when applying to FS on high-dimensional data. This limitation hinders PSO's applications on problems with hundreds of thousands of features, which become more popular in recent years. Furthermore, representation is the main factor in defining the size of the search space. An effective and flexible representation can help PSO achieve better solutions. In this paper, we propose the first PSO-based FS algorithm with a *variable-length representation*, so called VLPSO.

Different particles can have different lengths (i.e., numbers of features). Therefore, they may focus on different areas of the search space. Based on this ability, we propose a new initialization method called *population division*, which divides the population into divisions of particles with different lengths to provide an appropriate level of diversity for the whole population.

Furthermore, to encourage the short-length particles to find good feature subsets, we rearrange features in the descending order of their importance or relevance to the target concept. In other words, features are ranked based on a *feature ranking* method before applying PSO. In this way, the most important features can always be selected by any particle in the swarm. On the other hand, particles with longer lengths will have the potential to include less relevant features, enabling PSO to detect possible feature interactions that can lead to better feature subsets.

To facilitate particles with different lengths to learn from each other, VLPSO adopts the updating mechanism proposed in the comprehensive learning PSO (CLPSO) [14], which is a PSO variant. By allowing any particle to become an exemplar for others to learn from, CLPSO encourages diversity of the swarm and eliminates the need for specifying a specific communication topology. Furthermore, different dimensions of a particle can also learn from different particles. These characteristics of CLPSO enable our variable-length particles to choose appropriate exemplars easier. CLPSO has achieved significantly better results than many other PSO variants on many complex multimodal functions [14], [15]. However, to the best of our knowledge, CLPSO has never been applied to FS. In this paper, we will apply CLPSO to FS with some adjustments. First, the *exemplar assignment* in CLPSO needs to be adjusted to suit the newly proposed representation. Additionally, the probability used to choose exemplars in CLPSO is based on the index of the particle, which may limit its performance. Therefore, VLPSO will use an *adaptive learning probability* recently proposed in [15] to overcome this limitation.

Furthermore, we propose a *length changing* mechanism to alleviate the premature convergence, which is a common problem of PSO, especially on high-dimensional data. This mechanism enables particles to change length during the evolution. It helps PSO escape from local optima and move to more fruitful areas in the search space.

Additionally, PSO is well-known with the capability of quickly detecting fruitful regions; however, once there, it may not perform a refined local search well in complex search space to compute the optimum with high accuracy [16]. Local search has been combined with PSO to overcome this drawback [3]. Therefore, we also combine VLPSO with *local search* to further improve VLPSO performance on high-dimensional data.

Besides search mechanism, feature evaluation is another critical component of an FS method.

Although wrapper methods usually obtain better classification performance than filters, using classification accuracy solely may not be sufficiently powerful to distinguish the subtle difference between feature subsets. When working on high-dimensional data, classification algorithms require a much larger number of instances to maintain their performance due to the curse of dimensionality, which is usually not satisfied in reality. Therefore, we combine the strengths of both wrapper and filter approaches aiming to provide a smoother fitness landscape to facilitate the search process.

To avoid adding a notable amount of computation, we use *a hybrid evaluation method* of K-nearest neighbor (KNN) and a distance measure proposed in [3]. Since both use on the same distance measure in their calculations, the increase in evaluation time is neglectable.

B. Goals

The main goal of this paper is to propose the first variable-length representation in PSO for effective and efficient FS. Specifically, we will investigate the following research objectives.

1) How to design particles with different lengths that can communicate smoothly with each other.
2) Whether the feature subsets selected by the proposed algorithm are smaller and achieve similar or better classification performance than the original feature sets, and the subsets selected by standard PSO and other compared PSO-based FS methods.
3) Whether incorporating a local search procedure helps the proposed method achieve even higher classification accuracy.
4) Whether the proposed methods significantly reduce the running time of PSO on high-dimensional data.
5) Whether the proposed methods outperform traditional FS methods.
6) How effective and efficient the proposed strategies help PSO improve its performance on high-dimensional data.

II. Background and Related Work

A. Particle Swarm Optimization

As a population-based algorithm, PSO [9] maintains a swarm of particles. Each particle has a position which represents a candidate solution and a velocity showing the speed and direction that the particle should move in the next iteration. A particle's position and velocity are encoded in two vectors of n real numbers where n is the dimensionality of the problem. Particle's position is evaluated based on a fitness function. Then the best position that each particle has explored so far (pbest) is recorded and shared with other particles. In the conventional PSO, a fully connected topology is used to find the best position of the whole population (gbest). In other topologies which do not connect all particles, gbest is replaced with a local best (lbest). These best positions are used to update a particle's velocity which then defines its position as shown

$$v_{\mathrm{id}}^{t+1} = w * v_{\mathrm{id}}^{t} + c_1 * r_{1i} * (p_{\mathrm{id}}^{t} - x_{\mathrm{id}}^{t}) + c_2 * r_{2i} * (p_{\mathrm{gd}}^{t} - x_{\mathrm{id}}^{t}) \quad (1)$$

$$x_{\mathrm{id}}^{t+1} = x_{\mathrm{id}}^{t} + v_{\mathrm{id}}^{t+1} \quad (2)$$

where v_{id}^{t} and x_{id}^{t} are the velocity and position of the ith particle in dimension d at time t. w is the inertia weight representing the moving momentum of the particles. p_{id}^{t} and p_{gd}^{t} are pbest and gbest positions in dimension d at time t. c_1 and c_2 are acceleration constants, and r_1 and r_2 are random values uniformly distributed in [0, 1].

When applying PSO to FS, each real value ranging from 0 to 1 in the position vector indicates whether the corresponding feature should be selected or not based on a predefined threshold (e.g., 0.6).

B. Comprehensive Learning PSO

CLPSO proposed by Liang *et al.* [14] is a variant of continuous PSO in which a particle can learn from pbest of any particle. This strategy helps PSO maintain the diversity of the swarm and hence alleviate the common problem of premature convergence in PSO. Furthermore, while in standard PSO, all dimensions of a particle's velocity are updated based on its pbest and gbest, CLPSO enables different dimensions to learn from pbest of different particles including its own. The decision of choosing itself or another particle as an exemplar depends on a learning probability called Pc ranging from 0.05 to 0.5. Each particle has its own Pc. Equation (3) is used to calculate Pc for the ith particle

$$\mathrm{Pc}_i = 0.05 + 0.45\frac{\exp^{\frac{10(i-1)}{S-1}}}{\exp^{10}-1} \quad (3)$$

where S is the population size.

To set an exemplar for a dimension d of the ith particle, CLPSO generates a random number. If this number is greater than Pc_i, d learns from its own pbest; otherwise, a tournament selection with the size of 2 will be used to choose the exemplar for d. Therefore, besides a position and a velocity vector, CLSPO has another vector to record exemplars, which are the indexes of the chosen particles, for all dimensions. Exemplars of a particle remain unchanged until it stops improving for α iterations. Therefore, each particle also counts how many iterations that its pbest has not been changed. When this count exceeds α, all exemplars of this particle are renewed. With these changes, CLPSO uses (4) to update velocity

$$v_{\mathrm{id}}^{t+1} = w * v_{\mathrm{id}}^{t} + c * r_{\mathrm{id}} * (p_{\mathrm{exmplr}(id)d}^{t} - x_{\mathrm{id}}^{t}) \quad (4)$$

where exmplr(id) returns the exemplar of particle i in dimension d.

C. PSO for FS in Classification on High-Dimensional Data

PSO has been proposed and shown promise in FS [17]. An increased interest in PSO has shown through a growing number of papers proposing PSO-based FS methods in the past ten years [12].

Both filter or wrapper approaches have been proposed in PSO-based FS methods. In filter methods, different measures were proposed to evaluate feature subsets, such as rough set [18], fuzzy consistency [19], mutual information, and entropy [20]. On the other hand, feature evaluation methods in wrappers are based on the classification performance of a learning method [17]. Combination of both approaches has also been proposed [3].

To improve PSO performance for FS, researchers have also proposed many improvements in updating mechanisms of gbest [17], pbest [3] and particles [21], [22], communication topology [23], initialization [17], and representation [24]. Readers are referred to [12] for more comprehensive survey. In this section, we only focus on reviewing those methods that improve PSO representation.

Reducing PSO search space by explicitly eliminating redundant features is an effective way to improve PSO performance. Lane *et al.* [24] used a statistical clustering method to group similar features into the same cluster. Then, during the evolutionary process, some features with the highest probability (i.e., velocity) from each cluster were selected. Results showed that these methods could select a smaller number of features to achieve similar or better classification performance than all features and the compared methods. However, it is not easy to choose an appropriate number of features that should be selected from each cluster.

Among the early PSO variants, binary PSO [25] seems to significantly reduce the search space over continuous PSO (and also the memory space) when it restricts the position vector into binary values. However, using velocity solely to update its position, binary PSO cannot achieve a good performance [26]. Different updating mechanisms have been proposed to improve its performance [22], [26]. Nevertheless, these methods still use a fixed-length representation, which cannot scale well when the number of features reaches hundreds of thousands or even millions.

To provide a better solution for discrete optimization problems, Chen *et al.* [27] proposed a set-based PSO method (S-PSO) in which a particle is encoded as a crisp set of elements. Velocity is a set of elements and their corresponding possibilities. Results on the traveling salesman and multidimensional knapsack problems showed that S-PSO and its variants [28] were promising in solving discrete problems. However, the position and velocity representations in S-PSO require even more memory than standard PSO.

Another set-based representation (named SBPSO) was proposed by Langeveld and Engelbrecht [29], where the position is a set of elements while velocity is a set of operation pairs representing adding or removing elements. To avoid early convergence, velocity is updated using not only the pbest and lbest sets but also two more sets generated by two proposed operators. One is a random removal of some elements that appear in current position, pbest, and lbest sets. Another operator is adding some elements that are not in these three sets using a k-tournament selection that involves further fitness evaluations, which may lead to much higher computation time if the fitness function is costly. Results on knapsack problems showed that SBPSO performed significantly better than other three discrete PSO algorithms. However, the sensitivity analysis of SBPSO showed that its performance is sensitive to parameters used in the velocity formula [29].

In summary, although many PSO-based FS methods have been proposed to improve PSO performance in FS, not many studies addressed the limitation of the PSO representation in solving FS [12]. A new representation that can improve the scalability of PSO for FS, especially on high-dimensional data, is still missing. In the following section, we will propose a new approach to tackling the limitation of the fixed-length representation in PSO for FS.

III. Proposed Method

This section starts with a description of the proposed variable-length representation along with the exemplar assignment and an adaptive learning probability that are adjusted based on CLPSO. It then introduces the population division, feature ranking, and length changing strategies that are enabled by the variable-length representation. Finally, it presents the hybrid fitness function, the overall algorithm of VLPSO and the local search that is combined with VLPSO.

A. Variable-Length PSO Representation

The proposed variable-length representation aims to improve the scalability of PSO for FS on higher dimensional problems and reduce the computation time required when using a fix-length PSO method for FS.

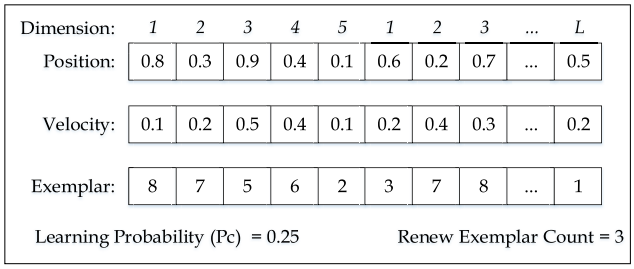

Fig. 1. Representation of a VLPSO particle with length L.

The proposed representation is still vector-based as the traditional PSO; however, each particle will have a different length L. VLPSO is developed based on the CLPSO [14] which was described in Section II-B. Fig. 1 shows the representation of a VLPSO particle with length L, which has three vectors including the position, velocity, and exemplar. Two additional variables record its learning probability (Pc) and the renew exemplar count (i.e., the number of iterations that pbest has not been improved).

The velocity and position updating in VLPSO follow (4) and (2), respectively.

B. Exemplar Assignment

In the original CLPSO, any particle can be used as an exemplar for a dimension of any particle. However, since different VLPSO particles have different lengths, the exemplar chosen for a particular dimension needs to have the corresponding dimension. In other words, the exemplar's length must exceed that dimension. Therefore, we propose a new exemplar updating mechanism for VLPSO, whose pseudocode is shown in Algorithm 1. The main difference between this method with the original one in CLPSO is that the two exemplars (p_1 and p_2) are randomly sampled (lines 9 and 10) until both of them satisfy the above-mentioned condition. However, if this condition is not met after a number of attempts, its own pbest will be used as the exemplar. In this method, the number of attempts is simply set to the population size.

C. Adaptive Learning Probability

In the original CLPSO, the probability of choosing exemplars for each dimension of a particle (Pc) is set based on its identity or index in the population and kept unchanged during the evolutionary process. As can be seen from (3), particles with a smaller index will have smaller Pc than those with a greater index. Therefore, according to the use of Pc for choosing exemplars in CLPSO, it is more likely that the small-index particles will follow its own pbest. However, to locate a better position or solution, particles should learn from particles with better fitness. Therefore, only those particles with better fitness should have smaller Pc, so that they can continue to exploit their good direction to find better pbest. In contrast, the worse particles should learn from the better ones. With this rationale, Yu *et al.* [15] proposed to calculate Pc for a particle based on its fitness rank instead of its index. As shown in (5),

Algorithm 1: Exemplar_Assignment

```
input : Particle i
output: Exemplar for each dimension of Particle i
begin
    L ← the length of particle i;
    for d = 1 to L do
        Rnd ← a random value drawn from a uniformly distribution;
        Pc_i ← Pc of particle i;
        if (Rnd >= Pc_i) then
            exemplar[d] ← i;
        else
            p_1 ← randomly pick a particle that is different from i and has a length longer than d;
            p_2 ← randomly pick a particle that is different from i and p_1 and has a length longer than d;
            if (p_1.fitness is better than p_2.fitness) then
                exemplar[d] ← p_1;
            else
                exemplar[d] ← p_2;
            end
        end
    end
    Return exemplar;
end
```

the proposed strategy assigns a smaller Pc for a particle with a smaller rank (i.e., a fitter particle). In this paper, we adopt this strategy since it has shown promise in improving the performance of CLPSO for function optimization [15]

$$\mathrm{Pc}_i = 0.05 + 0.45\frac{\exp^{\frac{10(\mathrm{rank}(i)-1)}{S-1}}}{\exp^{10}-1} \tag{5}$$

where S is the population size and rank(i) is the rank of particle i. The best particle in the swarm will be ranked 1.

D. Population Division

Based on the variable-length representation, VLPSO enables particles in the population to have any length that is smaller than the dimensionality of the problem. However, this may degrade PSO performance since particles will not learn much from each other when they are too different. Therefore, instead of setting a different length for each particle, we divide the whole population into a predefined number of divisions. In this way, we divide the search space into smaller subspaces, which can improve PSO's performance, especially in such a large and complex search space as in high-dimensional data.

The number of particles (or size) of each division (DivSize) is calculated based on the population size (PopSize) and the number of divisions (NbrDiv) as shown in (6). Particles in the same division will have the same length. The length of particles in a division V (ParLen_V) is calculated based on (7), where the maximum length (MaxLen) is the dimensionality of the problem. Note that within a division, particles with the same length can represent different feature subsets with different feature subset sizes. For example, two particles of length 8 represent two solutions, 10 100 001 and 10 001 111, which show which features are selected from the first eight features of the given dataset. They are corresponding to two

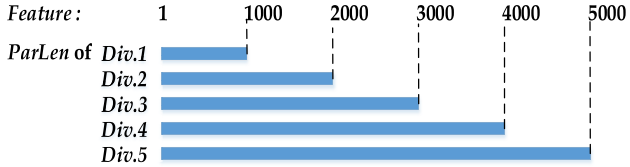

Fig. 2. Example of population division for a problem with 5000 features and the number of division is 5.

feature subsets, $\{F1, F3, F8\}$ and $\{F1, F5, F6, F7, F8\}$ which have the feature subset size of 3 and 5, respectively. Note that the dth dimension always represents the dth feature in all particles. This enables particles to learn from each other despite of their different lengths

$$\mathrm{DivSize} = \frac{\mathrm{PopSize}}{\mathrm{NbrDiv}} \tag{6}$$

$$\mathrm{ParLen}_V = \mathrm{MaxLen} * \frac{V}{\mathrm{NbrDiv}}. \tag{7}$$

Fig. 2 shows an example of the particle lengths in a problem with 5000 features and the number of divisions is set to 5. The length of all particles in Division 1 will be 1000. They will search for good feature subsets in the first 1000 features. Similarly, Division 2 will focus on feature subsets in the first 2000 features.

E. Feature Ranking

To rearrange features in the descending order of their relevance, we can use any measure to evaluate features. In this paper, we use the symmetric uncertainty (SU) [30] since it is a nonparametric measure and commonly used in FS methods [6], [31]. SU is a normalized version of information gain (IG) to evaluate feature relevance. To rank features, we use SU as shown in (8) to measure the correlation between a feature F and the class label C. The higher a feature correlates to the class label, the better it is

$$\mathrm{SU}(F, C) = \left[\frac{\mathrm{IG}(F|C)}{H(F) + H(C)}\right] \tag{8}$$

$$\mathrm{IG}(F|C) = H(F) - H(F|C) \tag{9}$$

where $H(F)$ is the entropy of F and $H(F|C)$ is the conditional entropy of F given C. The value of $\mathrm{SU}(F, C)$ ranges from 0 to 1, where 1 represents the most relevant feature.

F. Length Changing

During the evolutionary process, to help PSO jump out of possible local optima, we propose a length changing mechanism to direct particles to more fruitful areas in the search space, enabling PSO to reach better solutions in a shorter time. Particularly, when gbest does not change for a predefined number of iterations, we calculate the average fitness of all particles in each division and resize the particles to scale PSO search into the best division. In other words, the particle length of the best division will become the maximum length of the swarm after length changing. During this process, we keep the particles in the best division unchanged and resize those in the other divisions to a shorter length than the new maximum length. The process automatically changes the particles'

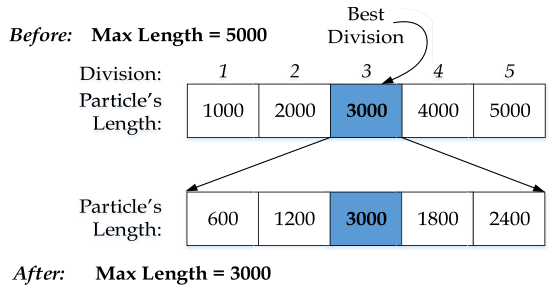

Fig. 3. Example of length changing in a swarm with five divisions.

length by cutting or appending more dimensions at the end of the representation while keeping the learned knowledge in the other dimensions. The number of dimensions being cut or appended is dynamically calculated based on the new length and the current length.

Fig. 3 shows a demonstration of this process on a swarm with five divisions. Initially, particle length of division 1, 2, 3, 4, and 5 are 1000, 2000, 3000, 4000, and 5000, respectively. Suppose the third division is be the best division with the highest average fitness, it is kept unchanged and 3000 becomes the new maximum length of the swarm. However, particle length of division 1, 2, 4, and 5 will be changed to 600, 1200, 1800, and 2400, respectively. Therefore, the last 400, 800, 2200, and 2600 dimensions will be cut in particle representations of division 1, 2, 4, and 5, respectively.

Algorithm 2 shows the pseudocode of the length changing procedure. If the current length of particles in a division is shorter than the new length, then more dimensions will be appended and randomly initialized (lines 12–16); otherwise, the exceeding dimensions will be removed (lines 18–21).

Since the maximal length of particles is always getting shorter every time particles' length is changed, another benefit of this mechanism is a dramatically reduction in the PSO computation time.

This length changing is applied when gbest does not improved for a number of iterations (β). β should be large enough for PSO to converge, and small enough for PSO to avoid being stuck in local optima for too long. Therefore, we conducted a sensitivity analysis as described in Section IV-D to choose suitable values for β and the number of divisions.

Algorithm 2: Length Changing Procedure

```
   input : Current swarm
   output: New swarm
1  begin
2      NbrDiv ← Number of divisions of the current swarm;
3      MaxLen ← Max length of particles in the current swarm;
4      Calculate the average fitness of each division;
5      BestLen ← Particles' length of the best division;
6      if (BestLen ≠ MaxLen) then
7          k ← 1;
8          for each division v (different from the best division) do
9              NewLen ← BestLen × k ÷ NbrDiv;
10             if (Length of particles in v < NewLen) then
11                 Append more dimensions to particles in
                   dividion v to have NewLen dimensions;
12                 Randomly initialise the new dimensions;
13                 Calculate fitness of these particles;
14                 k = k + 1;
15             else
16                 if (Length of particles in v > NewLen) then
17                     Remove the last dimensions of particles in
                       dividion v to have NewLen dimensions;
18                     Calculate fitness of these particles;
19                     k = k + 1;
20                 end
21             end
22         end
23         Calculate Pc for all particles using Eq. (5);
24         Renew exemplar of particles;      // Algorithm 1
25     end
26 end
```

G. Fitness Function

To combine the strengths of the wrapper and filter methods without significantly increasing the computation time, VLPSO uses the fitness function proposed in [3] to combine the accuracy of KNN and a distance measure [32] using a weight (γ) as shown in (10). While the classification accuracy can measure the performance of the feature subset, the distance measure can approximate how far these features can separate instances of different classes and unite those of the same class

$$\text{fitness} = (\gamma \cdot \text{accuracy} + (1-\gamma) \cdot \text{distance}). \tag{10}$$

To deal with unbalanced data in many high-dimensional datasets, we used a balanced accuracy [33] calculated based on (11) for the first component in the fitness function. Leave-one-out cross validation on the training data is used to evaluate the performance of KNN

$$\text{balanced\_accuracy} = \frac{1}{c}\sum_{i=1}^{c} \text{TPR}_i \tag{11}$$

where c is the number of classes of the problem, and TPR_i is the true positive ratio or the proportion of correctly identified instances in class i. Since there is no bias to any specific class, the weight for each class is set to $1/c$.

The distance measure is calculated based on (12), which aims at maximizing the distance between instances of different classes (D_b) and minimizing the distance between instances of the same class (D_w)

$$\text{distance} = \frac{1}{1+\exp^{-5(D_b - D_w)}} \tag{12}$$

where

$$D_b = \frac{1}{M}\sum_{i=1}^{M} \min_{\{j|j\neq i, \text{class}(I_i)\neq\text{class}(I_j)\}} \text{Dis}(I_i, I_j) \tag{13}$$

$$D_w = \frac{1}{M}\sum_{i=1}^{M} \max_{\{j|j\neq i, \text{class}(I_i)=\text{class}(I_j)\}} \text{Dis}(I_i, I_j) \tag{14}$$

where M is the number of instances in the training set. The distance between two instances $\text{Dis}(I_i, I_j)$ can be measured based on any distance approximation methods. In this paper, we use Manhattan measure since it is preferable than Euclidean distance metric for high-dimensional data [34]. To appropriately

apply this distance measure, the training data is scaled to the range of [0,1].

H. VLPSO Overall Algorithm

Fig. 4 shows the flowchart of VLPSO algorithm. It has three inputs, the number of divisions (NbrDiv), the maximum iterations that pbest is not improved to renew exemplars (α) for a particle, and the maximum iterations that gbest is not changed (β) to change particles' lengths. VLPSO starts with rearranging features based on the feature ranking described in Section III-E. After that, it initializes all divisions in the first loop, then calculates the learning probability Pc and assigns exemplars for each particle. The second loop is the evolutionary process. It repeats until reaching the maximum number of iterations. During this evolution, if gbest is not improved for β times, length changing procedure is called. Pc is also adapted based on (5).

The computation time of the proposed method and the baseline methods can be divided into two parts: 1) time for PSO updating and 2) time for fitness evaluation, in which fitness evaluation usually accounts for a much larger portion between the two. The former can be calculated based on the number of iterations (I), the number of particles (P), and the length of the particles which is equal to the number of original features (N) for PSO and ECLPSO. By dividing the population into D divisions, each of which has P/D particles with the particle length of $1 * N/D$, $2 * N/D, \ldots,$ or $D * N/D = N$, the proposed method reduces the time for PSO updating by half. For the fitness evaluation, the computation time highly depends on the number of features selected by each particle, which is hard to estimate and dataset-dependent. However, in the worse case that all particles select all features, the proposed method still takes only half of the time required by the baseline methods due to its shorter particle lengths.

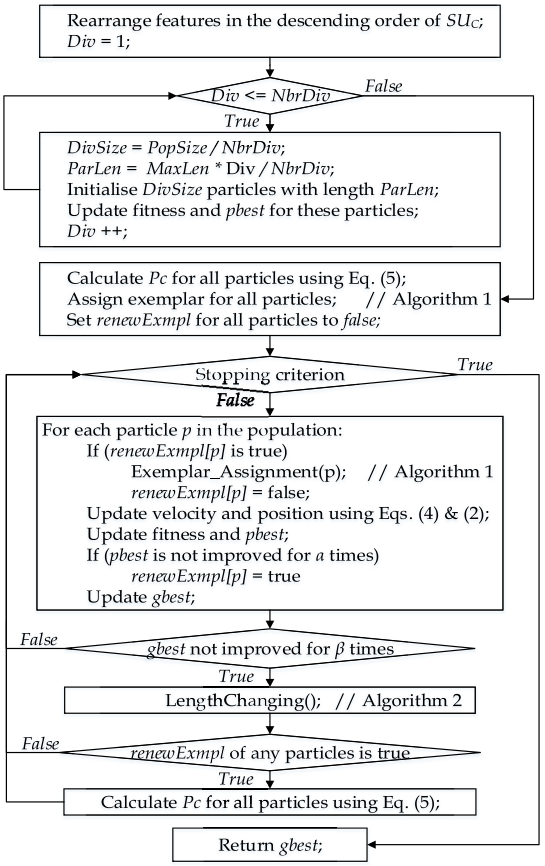

Fig. 4. Overall variable-length PSO algorithm.

I. VLPSO With Local Search

To further improve the performance of VLPSO on high-dimensional classification problems, we apply local search to pbest which was proposed in [3]. We call the combined method of VLPSO with local search as VLPSO-LS. This local search process aims to find a better solution surrounding the newly found pbest by randomly removing some redundant features and adding more relevant ones. The SU measure based on (8) is also used to evaluate feature relevance and redundancy in this process.

Given a binary vector corresponding to the feature subset of pbest, the local search procedure conducts a given number of tries. The more local search tries, the better solution can be found. Therefore, we set the number of tries to 100, which is equal to the maximum number of iterations PSO runs. However, thanks to the fast fitness evaluation used in the local search, 100 evaluations will not cost as much as in PSO. Each local search try considers to flip a random portion of pbest based on a given flipping size to create a new pbest. The size of the random portion is dynamically determined and proportional to the current pbest size. 25% is chosen to encourage removing more redundant features and adding more relevant ones in one local search try. The flipping process will scan features in this random portion to remove selected features if they are redundant and add nonselected features if they are relevant. A feature is defined as redundant if it is more correlated to other selected features than to the class label. A feature is relevant if it is more correlated to the class than the average correlation of all the selected features in the random portion. Therefore, the number of features actually flipped is not only dynamically determined by the current pbest size but also the characteristic of the dataset. As a result, the performance of the local search is not highly sensitive to the values of these two related parameters.

If a better pbest is found, it will replace the current one. Each local search try involves an evaluation process. Therefore, a significant number of evaluations will be added, which substantially increases the computation time. Therefore, to speed up VLPSO-LS, we also use the fast fitness evaluation strategy [3], which calculates the new distance between instances by adding to or subtracting from the current distance the value difference in features that are added or removed from the current pbest, respectively. Since a small portion of pbest is flipped in one local search try, this strategy saves a significant amount of computation, leading to much shorter evaluation time.

The frequency of applying local search can be predefined to compromise between its effectiveness and efficiency. When

TABLE I
DATASETS

Dataset	#Features	#Ins.	#Class	%Smallest class	%Largest class
SRBCT	2,308	83	4	13	35
Leukemia 1	5,327	72	3	13	53
DLBCL	5,469	77	2	25	75
9Tumor	5,726	60	9	3	15
Brain Tumor 1	5,920	90	5	4	67
Brain Tumor 2	10,367	50	4	14	30
Prostate	10,509	102	2	49	51
Leukemia 2	11,225	72	3	28	39
11Tumor	12,533	174	11	4	16
Lung Cancer	12,600	203	5	3	68

TABLE II
PARAMETER SETTINGS

Parameters	Settings
Population Size	#features/20 (restriction to 300)
Maximum iterations	100
$c1 = c2$ or c	1.49445
w	$0.9 - 0.5 * \frac{current\ iteration}{max\ iteration}$
Threshold for selected feature	0.6
Communication topology	Fully connected (PSO)
Max iterations for renew exemplars (α)	7 (ECLPSO, VLPSO, VLPSO-LS)
Local search tries	100 (VLPSO-LS)
Local search flipping size	25% of current $pbest$'s size (VLPSO-LS)
Number of divisions	12 (VLPSO, VLPSO-LS)
Max iterations for length changing (β)	9 (VLPSO, VLPSO-LS)

the trigger condition is satisfied, VLPSO-LS applies the local search procedure to the newly found pbest.

IV. EXPERIMENT DESIGN

A. Datasets

We used ten gene expression datasets with thousands of features that are publicly available on http://www.gems-system.org. Table I shows the number of features, instances, classes, and the percentage of instances in the smallest and largest class of each dataset. As can be seen from Table I, these datasets have a much smaller number of instances compared to the number of features. They are also highly unbalanced data. These characteristics make them become very challenging problems for both FS and classification.

B. Experiment Setting

Due to the small number of instances in these datasets, tenfold cross validation is used to create training and test sets for the experiments (no validation set involved). Onefold is kept as unseen test data, never used during the FS process. The remaining ninefold form the training data. Only the training data is used to during the FS process. After FS is finished, the training and test sets will be transformed based on the selected features and put into KNN to evaluate the performance of the FS method.

C. Baseline Methods

To evaluate the performance of VLPSO and VLPSO-LS, we compared the classification accuracy of KNN using the features selected by both methods with the original full feature sets, and the feature subsets selected by standard PSO (or PSO for short). They are also compared with the CLPSO enhanced with the adaptive learning probability [15] described in Section III-C, which we call ECLPSO for presentation convenience. All the compared methods will use the same fitness function and settings for common parameters. We also compare our methods with a recently proposed PSO-based FS method for high-dimensional classification using a competitive swarm optimizer (CSO) [21]. In this method, all particles are divided into two groups where pairwise comparison are applied and the better particle between the two will be used as an exemplar for the other. KNN is also used to evaluate feature subsets. We run the code provided by the authors on the same settings as other compared methods.

We also compared VLPSO-LS with three traditional FS methods, which are the linear forward selection (LFS), the correlation-based FS (CFS) [35], and the fast correlation-based FS method (FCBF) [6]. We chose these feature subset selection methods because of their popularity and the ability to automatically determine the number of selected features as our proposed methods. LFS is derived from the SFS where features are gradually added until no further improvement in classification accuracy. By restricting the number of features to be considered in each step, LFS [36] runs faster and finds smaller feature subsets with better classification performance than SFS. While LFS uses a wrapper approach, CFS is a filter FS method using the correlation measure to bias toward feature subset containing more relevant features and less redundant ones. Since best-first search is too expensive, especially on high-dimensional data, we ran CFS with a greedy forward selection. Unlike LFS and CFS, FCBF is a two-stage FS method where features are first ranked using the correlation measure and sorted in the descending order of relevance. Then a heuristic search is used to scan the ordered list to remove redundant features. Weka [37] was used to run the three methods with their default settings.

D. Parameter Settings

Table II shows the parameter settings used in the experiments. As can be seen from Table I, the datasets have very different numbers of features ranging from 2000 to 12 000. To deal with the large difference in the search space of different datasets, we set the population size to one twentieth of the number of features, but limited to 300 due to limited memory for computation. The parameter settings for the local search in VLPSO-LS are the same as in [3]. The maximum iterations for renewing exemplars (α) is set to 7 as suggested in [14]. The threshold for selected feature is usually set to 0.5 or slightly larger [17], [38]. Within a reasonable range, e.g., [0.5, 0.7], the value of this parameter does not significantly influence the selection process as investigated in [38]. The reason is during

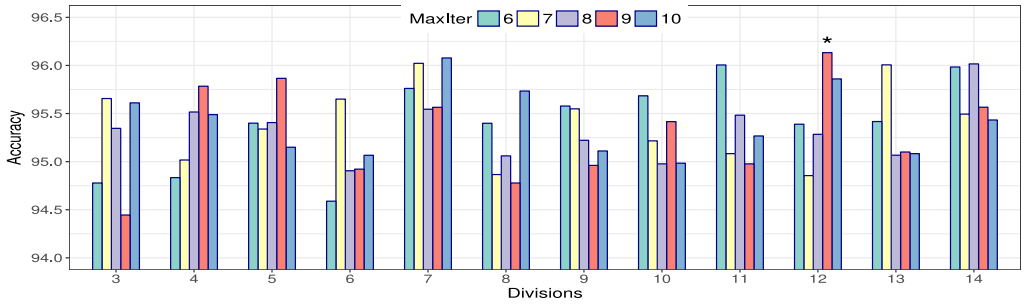

Fig. 5. Average results of 60 combinations of two parameters (number of divisions and maximum iterations to change particles' lengths).

the evolutionary process PSO can automatically update the particles' position values to make a feature become selected or not. PSO position updating is guided by fittest particles with better feature subsets. Therefore, PSO can adjust particle position values to make them higher or lower than the given threshold so that the feature subsets can obtain better fitness. We chose 0.6 to slightly prefer a smaller number of features at the early stage of the evolution.

The numbers of divisions and maximum iterations that gbest stays unchanged before changing particles' lengths (β) are new parameters that were proposed for variable-length PSO. Therefore, we conducted an experiment (sensitivity analysis) to find the appropriate values of these two parameters. The DLBCL dataset was used in this experiment because it has a medium size compared with other datasets. VLPSO-LS was run with 12 different values for the number of divisions ranging from 3 to 14, and 5 values from 6 to 10 for the maximum iterations to change particles' lengths, resulting in 60 combinations of these two parameters. Each combination was run 30 times. Fig. 5 shows the average test results of each combination with the best accuracy marked with an asterisk (*). The best combination was {12 divisions, 9 iterations} which VLPSO and VLPSO-LS used for all datasets.

Since PSO is a stochastic algorithm, 30 independent runs of each method with 30 different seeds are executed on each training set. As a result, each PSO method is run 300 times (30 runs $\times$ 10 folds) for each dataset. The average classification accuracies are reported and compared using Wilcoxon statistical significance test [39], with a 5% significance level. Experiments were runs on PC with Intel Core i7-4770 CPU@3.4 GHz and a total memory of 8 GB.

V. Results and Discussion

Table III shows the best and average test accuracy of KNN using the original feature set ("Full"), and the feature subset returned by the four PSO-based FS methods on each dataset. The reported accuracy is the balanced accuracy calculated using (11). The third and fourth columns show the running time (in minutes) and feature subset sizes. The smallest running time, size, and the highest average accuracy obtained on each dataset are bold. Columns S_1 and S_2 display the Wilcoxon significance test results (with a significance level of 0.05) of the corresponding method over VLPSO and VLPSO-LS, respectively. "+" or "−" means the result is significantly better or worse than the proposed method and "=" means they are similar in the Wilcoxon tests. In other words, the more −, the better the proposed methods.

A. VLPSO Results

1) VLPSO Versus Full: As can be seen from Table III, the numbers of features selected by VLPSO on all datasets were one to two orders of magnitude smaller than the original size with the best ratio of 1/398 in Prostate. Among all the compared methods, VLPSO obtained the smallest feature subsets on almost all datasets. With the smallest size, feature subsets returned by VLPSO significantly improved the performance of KNN on eight out of ten datasets. The highest improvement was seen on 9Tumor with 18.4% increase on average and 25% in the best case. On SRBCT, the proposed method selected less than 50 features to achieve 100% accuracy in almost all 300 runs, which is more than 12% improvement on Full. On Brain1, VLPSO selected about 27 out of 5920 features to obtain a similar classification performance as Full on average and 7% higher accuracy in the best case.

2) VLPSO Versus Standard PSO: Although PSO reduced the original feature sets by half, VLPSO still selected at least an order of magnitude fewer features than PSO on all datasets and achieved significantly better performance than PSO on nine datasets with the highest improvement of 12.7% on Leuk1. The highest dimensionality reduction was seen in Prostate where VLPSO selected 197 times fewer features than PSO and still improved 3.8% on the average PSO performance. Only on Brain1, VLPSO obtained 2.5% lower average accuracy than PSO while selected 109 times fewer features. However, the best accuracy achieved by VLPSO on Brain1 is still 2% higher than PSO.

3) VLPSO Versus ECLPSO: Although ECLPSO selected a smaller number of features than PSO on all datasets, its performance was quite similar to PSO with a maximum 2% difference in accuracy. Compared with ECLPSO, VLPSO also selected a much smaller number of features and achieved significantly better performance on all datasets except for Brain1.

4) VLPSO Versus CSO: Compared with VLPSO, CSO selected more features on nine datasets. On Prostate, CSO selected 357 features while VLPSO selected 26 features only.

TABLE III
AVERAGE TEST RESULTS

Dataset	Method	Time(m)	Size	Best	Mean ± Std	S_1	S_2
SRBCT	Full		2308.0	87.08		-	-
	PSO	8.2	1119.4	92.50	89.51 ± 1.56	-	-
	ECLPSO	7.5	1054.8	90.42	88.10 ± 1.57	-	-
	CSO	19.9	85.4	100.00	93.29 ± 3.52	-	-
	VLPSO	**1.4**	**49.1**	100.00	99.67 ± 0.52		=
	VLPSO-LS	2.1	71.4	100.00	**99.75 ± 0.45**		
DLBCL	Full		5469.0	83.00		-	-
	PSO	47.6	2681.0	86.33	83.67 ± 1.52	-	-
	ECLPSO	44.2	2491.3	86.33	82.44 ± 2.01	-	-
	CSO	394.8	**30.1**	100.00	94.30 ± 4.05	+	=
	VLPSO	**7.4**	48.1	93.33	86.51 ± 2.88		-
	VLPSO-LS	8.8	59.9	99.17	**96.13 ± 1.90**		
9Tumor	Full		5726.0	36.67		-	-
	PSO	39.2	2811.9	45.00	42.72 ± 1.42	-	-
	ECLPSO	39.2	2605.5	45.00	41.33 ± 1.48	-	-
	CSO	373.4	220.3	68.33	**59.50 ± 3.72**	+	+
	VLPSO	**6.2**	**44.2**	61.67	55.11 ± 4.71		=
	VLPSO-LS	**6.2**	61.9	70.00	56.78 ± 5.23		
Leuk1	Full		5327.0	79.72		-	-
	PSO	41.2	2615.5	87.36	80.60 ± 2.55	-	-
	ECLPSO	36.3	2427.9	87.64	80.88 ± 2.28	-	-
	CSO	251.8	170.1	96.81	88.45 ± 3.90	-	-
	VLPSO	**6.4**	**54.7**	97.92	93.31 ± 2.34		=
	VLPSO-LS	7.9	59.3	95.42	**93.75 ± 1.56**		
Brain1	Full		5920.0	72.08		=	-
	PSO	66.7	2917.2	77.08	73.73 ± 2.21	+	-
	ECLPSO	60.0	2710.0	77.08	73.87 ± 2.37	+	-
	CSO	462.1	207.6	86.67	**79.93 ± 3.09**	+	+
	VLPSO	**9.8**	**26.8**	79.17	71.19 ± 3.52		-
	VLPSO-LS	13.0	102.1	81.25	75.54 ± 2.79		
Leuk2	Full		11225.0	89.44		-	-
	PSO	120.6	5535.7	92.22	89.83 ± 1.00	-	-
	ECLPSO	125.6	5115.6	92.22	89.82 ± 1.20	-	-
	CSO	1845.2	88.6	98.33	91.72 ± 3.16	=	-
	VLPSO	**16.9**	**35.2**	94.44	91.56 ± 1.67		-
	VLPSO-LS	18.3	61.2	96.67	**95.39 ± 0.95**		
Brain2	Full		10367.0	62.50		-	-
	PSO	80.5	5117.2	67.08	61.99 ± 2.91	-	-
	ECLPSO	73.6	4718.7	68.75	63.20 ± 2.60	-	-
	CSO	950.8	90.43	90.83	**80.44 ± 6.28**	+	+
	VLPSO	**12.1**	81.5	73.33	66.78 ± 4.10		-
	VLPSO-LS	13.7	**61.4**	82.92	73.25 ± 4.30		
Prostate	Full		10509.0	85.33		-	-
	PSO	160.6	5193.7	88.33	86.00 ± 1.49	-	-
	ECLPSO	152.5	4818.5	88.33	85.46 ± 1.41	-	-
	CSO	2369.9	357.2	95.17	88.99 ± 2.68	=	-
	VLPSO	**22.6**	**26.4**	94.17	89.82 ± 2.28		-
	VLPSO-LS	25.8	56.4	97.17	**92.58 ± 1.47**		
Lung	Full		12600.0	78.05		-	-
	PSO	574.2	6234.7	82.72	78.77 ± 1.53	-	-
	ECLPSO	503.1	5739.7	81.64	77.91 ± 1.98	-	-
	CSO	5565.9	226.4	93.79	87.72 ± 2.93	-	-
	VLPSO	**70.1**	**176.1**	94.08	89.47 ± 2.18		=
	VLPSO-LS	307.1	242.9	93.71	**90.17 ± 2.10**		
11Tumor	Full		12533.0	71.42		-	-
	PSO	418.5	6205.0	75.59	71.81 ± 1.75	-	-
	ECLPSO	366.7	5731.7	74.09	71.09 ± 1.20	-	-
	CSO	6288.6	588.6	84.47	79.52 ± 2.35	=	-
	VLPSO	**65.8**	**246.7**	85.16	80.81 ± 2.32		-
	VLPSO-LS	99.0	367.4	86.51	**82.81 ± 2.09**		

In terms of classification accuracy, VLPSO achieved a significantly better or similar classification performance as CSO on six datasets, and worse on the remaining four datasets.

In summary, VLPSO won 32, draw 4 and lost 4 out of the 40 comparisons in terms of classification performance while selecting the smallest feature subsets in almost all cases. Its results indicated that VLPSO conducted a much better search than the compared methods. VLPSO effectiveness is contributed by two mechanisms, the population division and length changing, which are enabled by using the variable-length representation to encode candidate solutions with different lengths. The population division distributes particles in the swarm into different areas of the search space, which effectively ensures the diversity of the swarm. Furthermore, when there is a sign of being stuck in local optima, the proposed length changing mechanism enabled particles to change their search space without throwing away the knowledge that they have learned so far. This mechanism also gradually adjusts PSO search to focus on smaller and more fruitful areas, enabling PSO to find much smaller feature subsets with better discriminating ability.

B. VLPSO-LS Results

1) VLPSO-LS Versus Full: As can be seen from Table III, the number of features selected by VLPSO-LS on all datasets was one to two orders of magnitude smaller than the original size. The features selected by VLPSO-LS helped KNN obtain significantly better accuracy than using Full on all datasets with an increase of more than 10% on seven datasets. On 9Tumor, VLPSO-LS subsets obtained 20% higher accuracy than Full on average and 33% higher in the best case.

2) VLPSO-LS Versus Standard PSO and ECLPSO: The results of significance test shown in Column S_2 showed that VLPSO-LS outperformed PSO on all datasets while selected 16 to 92 times smaller number of features. Seven out of ten datasets witnessed an increase of at least 10% on average accuracy with the highest improvement of 14% on 9Tumor.

3) VLPSO-LS Versus ECLPSO: Comparison between ECLPSO and VLPSO-LS yields a similar pattern as with PSO where VLPSO-LS selected 14 to 85 times smaller number of features than ECLPSO to achieve significantly better performance on all datasets.

4) VLPSO-LS Versus CSO: Although CSO selected a much smaller number of features than PSO and ECLPSO, its feature subsets were still up to 6.3 times larger than VLPSO-LS on eight datasets. VLPSO obtained significantly better classification performance than CSO on six datasets that had up to 6.4% higher average accuracy, similar on one and worse on the remaining three datasets, namely 9Tumor, Brain1, and Brain2.

5) VLPSO-LS Versus VLPSO: As shown in Table III, VLPSO-LS selected slightly more features than VLPSO on almost all cases to further improve the performance of VLPSO on six datasets. The highest improvement was on DLBCL with 9.6% higher accuracy. On Leuk2, VLPSO-LS selected 20 features less than VLPSO while increased VLPSO accuracy by 6.5% on average and 9.6% in the best case. While VLPSO obtained a similar or worse performance than Full and other PSO methods on Brain1, VLPSO-LS significantly outperformed the others on this dataset.

In summary, VLPSO-LS won 42, draw 5 and lost 3 out of 50 comparisons. The results of VLPSO-LS indicated that by removing redundant features and adding more relevant ones, the local search strategy helped VLPSO fine tune its solutions to achieve the highest accuracy on all the datasets.

C. Computation Time

As can be seen from the third column of Table III, the fastest algorithm among all the five compared methods is VLPSO. Although VLPSO-LS performed more fitness evaluations, it

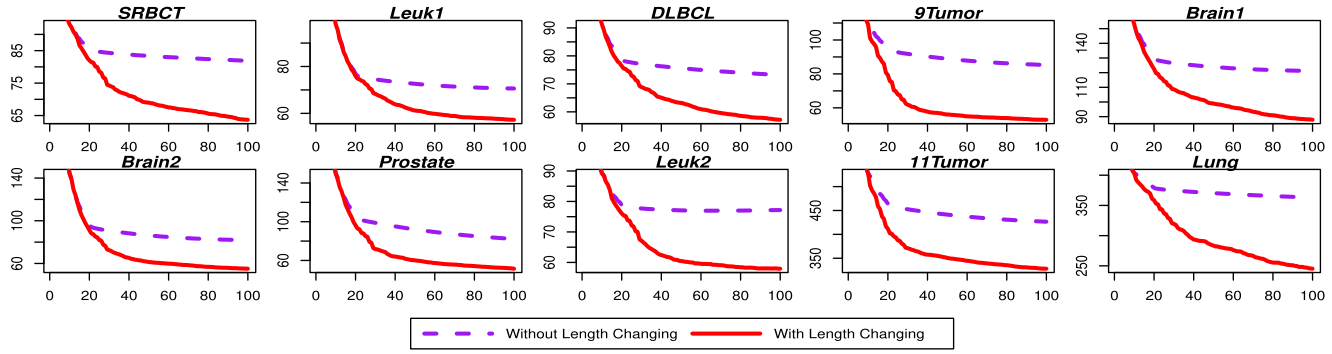

Fig. 6. Average feature subset size of the whole swarm from iteration 10 to 100 in both scenarios with and without using length changing.

is the second-fastest method with only a slightly longer time than VLPSO. PSO and ECLPSO, in third place, are similar with their running time 5 to 8 times longer than VLPSO on all datasets. Finally, CSO required the longest running time at 14 to 109 times longer than VLPSO. This may be due to the strategy of recording the historical fitness values of all previously selected feature subset in an archive to avoid re-evaluating the same solution. This strategy has been shown to be effective in [21] where the largest dataset had 1588 features. However, when the number of features increases to tens of thousands of features, the evaluation time saved seemed to be affected by the time needed for matching the archived solutions which used a fix-length representation with its length equal to the original number of features.

In summary, the variable-length PSO-based methods required a much shorter running time than the traditional fix-length ones. Section VI will further investigates the differences in evolutionary processes that contribute to the effectiveness and efficiency of the proposed methods.

D. Effect of Length Changing

To investigate the effect of length changing, we analyze the results of VLPSO-LS with (W) and without (WO) applying length changing mechanism, which is shown in Table V given in the Appendix. Compared to WO, W obtained up to 10 fewer features on all datasets except for Lung where it selected 1.8 more features on average. In terms of the classification accuracy, W results were significantly better than WO on DLBCL and similar on the remaining datasets. Furthermore, W saved up to 14% of WO running time on all datasets. Fig. 6 shows the average feature subset size of the whole swarm from iteration 10 (i.e., just after the first time length changing is applied in the evolutionary process) to 100 in both scenarios. The figure shows that length changing dramatically reduced particle lengths, which required a shorter time for PSO updating.

E. Comparisons With Traditional Methods

To see if the proposed methods performed better than the traditional FS methods, we compared the results of VLPSO-LS with those returned from LFS, CFS, and FCBF, all of which can automatically determine the number of features that should be selected. Table IV showed the running time, the returned feature subset size, and the best and mean accuracy of each method. Column *S* showed the results of Wilcoxon significance test compared the corresponding method over VLPSO-LS using the same symbols and meanings as in Table III. The smallest size, the highest average, and best accuracy obtained on each dataset are bold.

TABLE IV
VLPSO-LS VERSUS TRADITIONAL METHODS

Dataset	Method	Time (s)	Size	Best	Mean ± Std	*S*
SRBCT	LFS	25.0	**7.1**	91.67		-
	CFS	243.3	112.3	99.17		-
	FCBF	1.4	69.0	98.75		-
	VLPSO-LS	123.3	71.4	**100.00**	**99.75 ± 0.45**	
DLBCL	LFS	56.3	**5.9**	83.33		-
	CFS	778.4	86.3	93.00		-
	FCBF	1.6	66.1	94.83		-
	VLPSO-LS	527.3	59.9	**99.17**	**96.13 ± 1.90**	
9Tumor	LFS	52.9	**9.7**	26.67		-
	CFS	341.2	44.0	56.67		=
	FCBF	1.7	33.7	55.00		=
	VLPSO-LS	371.4	61.9	**70.00**	**56.78 ± 5.23**	
Leuk1	LFS	51.9	**5.4**	85.14		-
	CFS	778.4	79.4	92.08		-
	FCBF	1.4	48.5	89.86		-
	VLPSO-LS	471.5	59.3	**95.42**	**93.75 ± 1.56**	
Brain1	LFS	77.9	**12.2**	63.33		-
	CFS	2973.0	151.9	76.67		+
	FCBF	2.8	104.6	73.75		-
	VLPSO-LS	781.6	102.1	**81.25**	75.54 ± 2.79	
Leuk2	LFS	143.4	**4.7**	89.44		-
	CFS	5653.0	129.5	94.44		-
	FCBF	4.1	77.5	95.56		=
	VLPSO-LS	1099.1	61.2	**96.67**	95.39 ± 0.95	
Brain2	LFS	113.9	**9.1**	77.50		+
	CFS	3182.2	101.1	77.50		+
	FCBF	2.7	66.2	77.50		+
	VLPSO-LS	820.6	61.4	**82.92**	73.25 ± 4.30	
Prostate	LFS	158.2	**5.9**	90.17		-
	CFS	2537.4	80.4	92.17		=
	FCBF	3.4	66.1	92.17		=
	VLPSO-LS	1550.5	56.4	**97.17**	**92.58 ± 1.47**	
Lung	LFS	358.8	**8.5**	79.62		-
	CFS	85179.1	517.0	**93.76**		+
	FCBF	56.7	439.4	92.71		+
	VLPSO-LS	18425.0	242.9	93.71	90.17 ± 2.10	
11Tumor	LFS	309.3	**17.3**	61.76		-
	CFS	57340.7	361.6	80.04		-
	FCBF	31.1	349.6	80.57		-
	VLPSO-LS	5941.5	367.4	**86.51**	**82.81 ± 2.09**	

1) VLPSO-LS Versus LFS: As can be seen from Table IV, LFS selected less than 20 features on all datasets, obtaining the smallest feature subset of all the compared methods. However, these smallest subsets obtained significantly lower accuracy than VLPSO-LS on nine datasets with more than 10% difference on five cases. On 9Tumor, VLPSO-LS selected 52 more

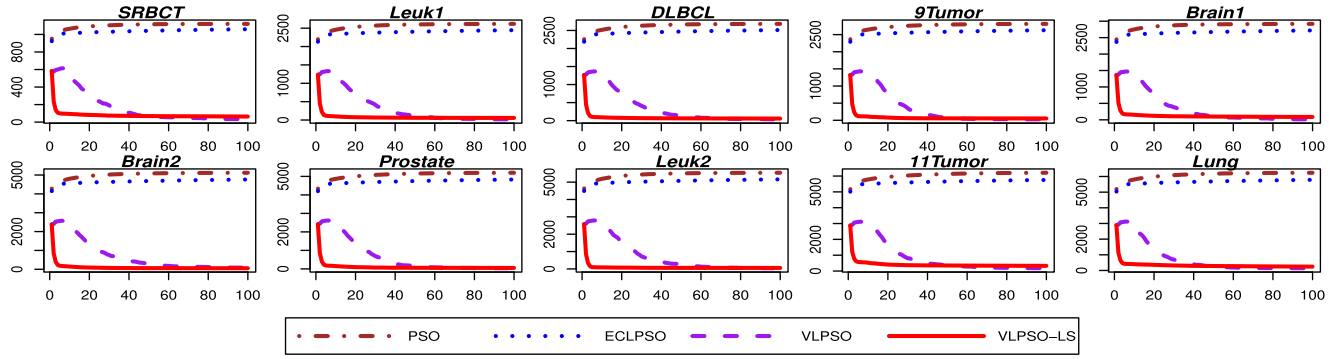

Fig. 7. Average feature subset size of the whole swarm in 100 iterations.

features to achieve 30% higher accuracy than LFS on average and 43% higher in the best case. Only on Brain2, VLPSO-LS obtained a lower average accuracy than LFS, but in the best case, VLPSO-LS still achieved more than 5% higher accuracy than LFS. The results indicate that the linear forward search in LFS was trapped in local optima in a very early stage, resulting in minimal but low performance feature sets.

2) VLPSO-LS Versus CFS: Compared with CFS, VLPSO-LS selected fewer features on eight datasets to obtain significantly better accuracies on five, and similar on two. Although both had a similar classification performance on 9Tumor, VLPSO-LS still obtained 14% higher accuracy in the best case. On Brain1 and Lung, VLPSO-LS had 1%–3% lower average accuracy; however, with much smaller feature subsets and could achieve better accuracy in the best case. We also note that although CFS is a deterministic and filter FS method, its running time is two times longer than VLPSO-LS on the small SRBCT dataset, and nine times on the large 11Tumor dataset. This indicates that VLPSO-LS better scale to high-dimensional data than CFS.

3) VLPSO-LS Versus FCBF: The fourth column of Table IV showed that the difference in feature subset size between VLPSO-LS and FCBF was very small on all datasets except Lung, where VLPSO-LS selected about a half of FCBF. With a similar size, feature subsets of VLPSO-LS obtained a significantly better accuracy than FCBF on five datasets, and similar on two. Selecting 11 more features on Leuk1, VLPSO-LS achieved 3.9% higher accuracy than FCBF on average and 5.6% higher in the best case. On Brain2, although VLPSO-LS had 4.3% lower average accuracy, its best accuracy is still $5.4 higher than FCBF. We also note that FCBF scaled very well to high-dimensional data and it is the fastest method among the four, which is an advantage of a filter and ranking FS method. However, the inferior results of FCBF suggested that the heuristic search of FCBF in the second stage might get stuck in local optima while the global search helped PSO overcome this problem to obtain better results.

In summary, among 30 comparisons with the three methods, VLPSO-LS won 19 cases, drew 6 and lost 5. The results showed that VLPSO-LS achieved a significantly better performance than the traditional methods in reasonable running time.

VI. Further Analysis

We have shown so far that in most cases, VLPSO and VLPSO-LS achieved much better performance than the compared FS methods in terms of classification accuracy, dimensionality reduction, and computation time. In this section, we will further investigate their performance to reveal the contributions of different components to improving PSO search capability. Specifically, we will investigate the effect of variable-length representation and the local search. Note that the results shown in all figures of this section are averaged over the 30 runs.

A. Efficiency of Variable-Length PSO Representation

First of all, we will investigate the effect of variable-length representation on the tremendous reduction of computation time. Since all the four PSO-based FS methods used the same population size, the maximum number of iterations, and the fitness function, their running time difference is contributed by the feature subset size which affected the fitness evaluation time and the length of particles which affected the particle updating time. To investigate these differences, we plot the average feature subset size of a particle in each iteration of all the four methods and the particles' maximum length of the two proposed methods in Figs. 7 and 8, respectively.

As can be seen from Fig. 7, the subset sizes of PSO and ECLPSO particles started quite large and slightly increased over the whole evolutionary process. On the other hand, starting at about half size of PSO, VLPSO witnessed a steady decrease in the first 40 iterations and then a slight decline in the remaining stage. Starting at the same point as VLPSO, VLPSO-LS dramatically dropped to a very small size in the first several iterations and kept stable till the end. These figures clearly showed that the fitness evaluation time in VLPSO and VLPSO-LS are significantly reduced thanks to the small subset sizes of all particles.

In addition to the small evaluation time, the particle updating time of the variable-length PSO is also much smaller. With a fix-length representation, PSO and ECLPSO spent a fixed amount of time to update particles with the length of the original number of features in every iteration. On the other

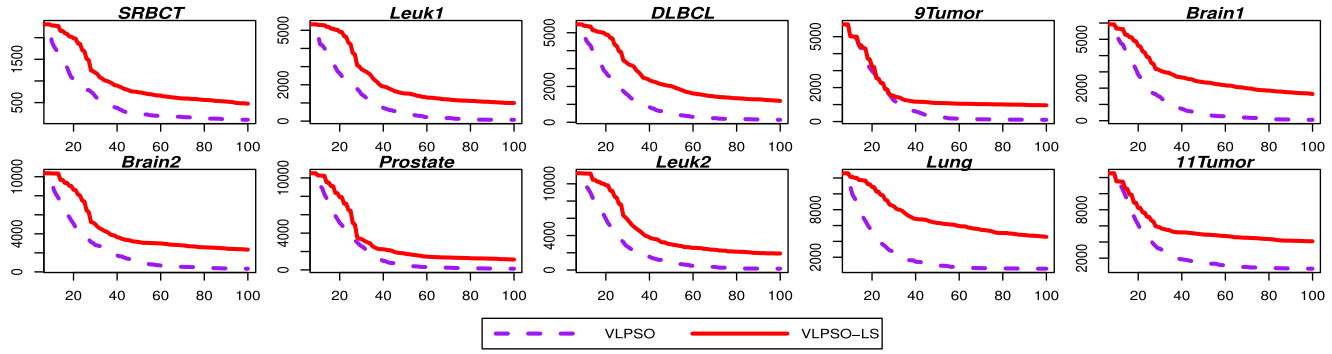

Fig. 8. Average particle maximum length in 100 iterations.

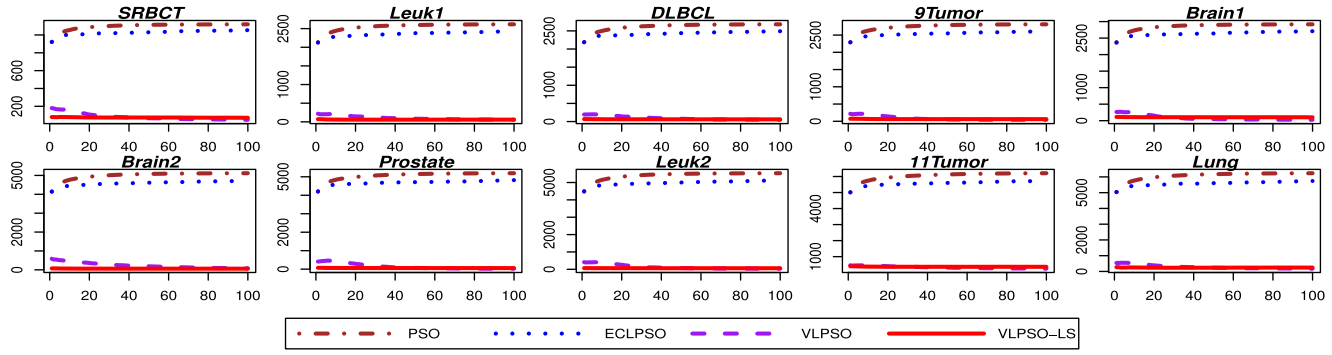

Fig. 9. Average feature subset size of gbest in 100 iterations.

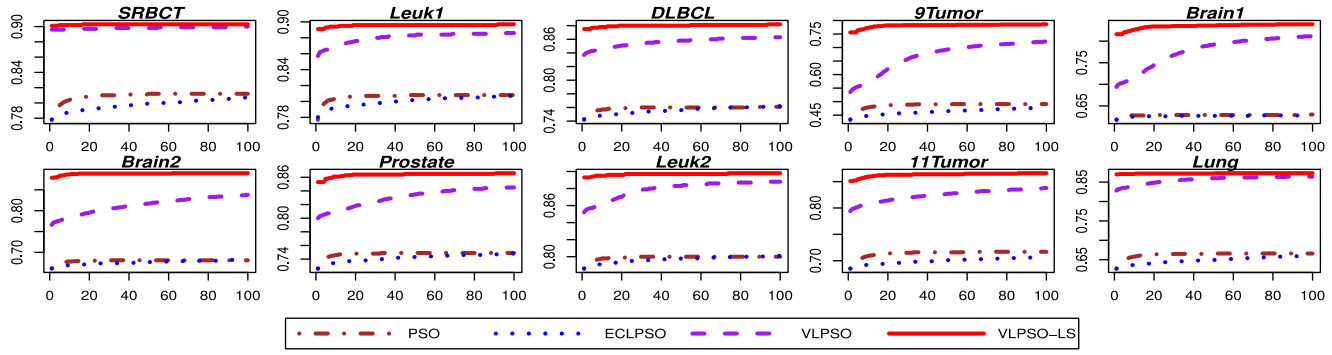

Fig. 10. Average fitness of gbest in 100 iterations.

hand, using the size division strategy, VLPSO and VLSPO-LS spent much shorter time to update their particles from the first iteration to the end. Furthermore, after each length changing, the particle lengths even get significantly shorter. Fig. 8 showed the maximum length of particles in VLPSO and VLSPO-LS changing during the evolutionary process of all datasets. The figures showed that the maximum length dramatically dropped in the first 40 iterations and slightly decreased after that. The significant impact of particle lengths on the running time can be shown in the Lung dataset. In this dataset, while the average feature subset sizes of VLPSO-LS were much smaller than VLPSO in the first 40 iterations as shown in Fig. 7, the particles' maximum length in VLPSO-LS is much larger than VLPSO as shown in Fig. 8, which makes VLPSO-LS had a much longer running time than VLPSO as shown in Table III.

B. Effectiveness of Variable-Length PSO Representation

Second, we will investigate the effect of variable-length representation on the size and accuracy of the returned feature subsets. Figs. 9 and 10 show the changing of gbest size and fitness during the evolutionary process. Note that the fitness values are calculated based on (10), which is a combination of KNN accuracy and the distance measure. Therefore, these values hardly reached the value of 1.

As shown in Fig. 9, from the beginning of the run, gbest size of the two variable-length PSO methods was already way smaller than the fixed ones. By dividing particles into different length divisions, the swarm in the proposed methods could have a higher diversity than the baseline methods. This enabled them to find much smaller feature subsets from the early stage of the evolutionary process. By removing redundant features from pbest, VLPSO-LS had an even smaller

TABLE V
COMPARED RESULTS OF VLPSO-LS IN BOTH SCENARIOS WITH (W) AND WITHOUT (WO) USING THE LENGTH CHANGING MECHANISM

Dataset	Method	Time(m)	Size	Best	Mean ± Std	S
SRBCT	WO	2.10	76.3	100.00	99.89 ± 0.29	=
	W	2.05	71.4	100.00	99.75 ± 0.45	
DLBCL	WO	9.14	62.5	98.33	95.03 ± 2.00	-
	W	8.79	59.9	99.17	96.13 ± 1.90	
9Tumor	WO	7.19	67.6	63.33	55.89 ± 3.93	=
	W	6.19	61.9	70.00	56.78 ± 5.23	
Leuk1	WO	8.74	61.8	97.64	93.63 ± 2.09	=
	W	7.86	59.3	95.42	93.75 ± 1.56	
Brain1	WO	13.39	107.6	82.08	76.15 ± 3.41	=
	W	13.03	102.1	81.25	75.54 ± 2.79	
Leuk2	WO	20.64	65.4	98.33	95.26 ± 1.37	=
	W	18.32	61.2	96.67	95.39 ± 0.95	
Brain2	WO	14.99	67.0	79.58	73.50 ± 3.55	=
	W	13.68	61.4	82.92	73.25 ± 4.30	
Prostate	WO	27.14	61.0	96.17	92.49 ± 1.93	=
	W	25.84	56.4	97.17	92.58 ± 1.47	
Lung	WO	312.96	241.1	94.19	90.14 ± 2.14	=
	W	307.08	242.9	93.71	90.17 ± 2.10	
11Tumor	WO	102.09	377.3	85.96	82.79 ± 1.84	=
	W	99.02	367.4	86.51	82.81 ± 2.09	

gbest size than VLPSO from the beginning and kept nearly unchanged to the end. On the other hand, VLSPO's gbest maintained a gradual decrease over the whole evolutionary process and reached a smaller subset size than VLPSO-LS at the end.

As shown in Fig. 10, with a small subset size, VLPSO's gbest obtained a much higher fitness than the baseline methods from the first iteration and continued to improve to the end. Using an informative local search to remove redundant features and add more relevant ones, VLPSO-LS's gbest even had much better fitness than VLPSO. The gap between gbest's fitness of both methods varied in different datasets; however, with the same trend which is getting closer at the end. The gap's magnitude may reflect the complexity of the corresponding search space. For example in SRBCT, a small dataset with 2038 features, this gap is quite small, and the significance test on 30 runs showed that both methods obtained a similar classification accuracy while VLPSO-LS selected 20 more features than VLPSO. On the other hand, on such datasets with a greater number of classes and features as 9Tumor and 11Tumor, the gap is quite large even at the end.

VII. CONCLUSION

This paper aims to propose a new PSO representation that can have a variable and dynamic length for FS on high-dimensional data. The goal was achieved by proposing a new variable-length PSO-based FS method, where particles in a swarm can have different lengths which can also be changed during the evolutionary process. The results showed that the proposed variable-length PSO-based methods achieved a much smaller feature subset with better classification performance in a shorter time than the traditional fixed-length methods. By having shorter and dynamic lengths to encode particles, PSO maintains a better diversity in the swarm and requires a much smaller number of updating operations. The proposed length changing mechanism also helped PSO jump out of local optima and focus its search on a more fruitful area.

The proposed variable-length PSO-based method has shown promise in FS. It can also be applied to other tasks. In the proposed representation, each dimension is updated separately without taking into account other dimensions. This may limit the performance of PSO in FS on problems which have strong interactions between features for target prediction. Taking this information into account when selecting features may help PSO obtain even better results, but it is very hard to achieve. This direction will be considered in our future work.

APPENDIX

See Table V.

REFERENCES

[1] M. Dash, "Feature selection via set cover," in *Proc. IEEE Knowl. Data Eng. Exchange Workshop*, Newport Beach, CA, USA, Nov. 1997, pp. 165–171.

[2] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[3] B. Tran, B. Xue, and M. Zhang, "A PSO based hybrid feature selection algorithm for high-dimensional classification," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 3801–3808.

[4] Y. Sun, "Iterative RELIEF for feature weighting: Algorithms, theories, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1035–1051, Jun. 2007.

[5] A. Jakulin and I. Bratko, "Testing the significance of attribute interactions," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, Banff, AB, Canada, 2004, pp. 52–59.

[6] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proc. 20th Int. Conf. Mach. Learn. (ICML)*, Washington, DC, USA, 2003, pp. 856–863.

[7] A. W. Whitney, "A direct method of nonparametric measurement selection," *IEEE Trans. Comput.*, vol. C-20, no. 9, pp. 1100–1103, Sep. 1971.

[8] T. Marill and D. M. Green, "On the effectiveness of receptors in recognition systems," *IEEE Trans. Inf. Theory*, vol. IT-9, no. 1, pp. 11–17, Jan. 1963.

[9] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, Nagoya, Japan, 1995, pp. 39–43.

[10] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 644–660, Aug. 2017.

[11] C. Yue, B. Qu, and J. Liang, "A multi-objective particle swarm optimizer using ring topology for solving multimodal multi-objective problems," *IEEE Trans. Evol. Comput.*, to be published, doi: 10.1109/TEVC.2017.2754271.

[12] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 606–626, Aug. 2016.

[13] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[14] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[15] X. Yu, Y. Liu, X. Feng, and G. Chen, "Enhanced comprehensive learning particle swarm optimization with exemplar evolution," in *Proc. 11th Int. Conf. Simulat. Evol. Learn. (SEAL)*, 2017, pp. 929–938.

[16] P. J. Angeline, *Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences*. Heidelberg, Germany: Springer, 1998, pp. 601–610.

[17] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms," *Appl. Soft Comput.*, vol. 18, pp. 261–276, May 2014.

[18] H. H. Inbarani, A. T. Azar, and G. Jothi, "Supervised hybrid feature selection based on PSO and rough sets for medical diagnosis," *Comput. Methods Programs Biomed.*, vol. 113, no. 1, pp. 175–185, 2014.
[19] B. Chakraborty and G. Chakraborty, "Fuzzy consistency measure with particle swarm optimization for feature selection," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, Manchester, U.K., 2013, pp. 4311–4315.
[20] R. Tahmasebifar, M. K. Sheikh-El-Eslami, and R. Kheirollahi, "Point and interval forecasting of real-time and day-ahead electricity prices by a novel hybrid approach," *IET Gener. Transm. Distrib.*, vol. 11, no. 9, pp. 2173–2183, Jun. 2017.
[21] S. Gu, R. Cheng, and Y. Jin, "Feature selection for high-dimensional classification using a competitive swarm optimizer," *Soft Comput.*, vol. 22, no. 3, pp. 811–822, 2018.
[22] H. Banka and S. Dara, "A hamming distance based binary particle swarm optimization (HDBPSO) algorithm for high dimensional feature selection, classification and validation," *Pattern Recognit. Lett.*, vol. 52, pp. 94–100, Jan. 2015.
[23] A. Moaref and V. S. Naeini, "A particle swarm optimization based on a ring topology for fuzzy-rough feature selection," in *Proc. 13th Iran. Conf. Fuzzy Syst. (IFSC)*, 2013, pp. 1–6, doi: 10.1109/IFSC.2013.6675598.
[24] M. Lane, B. Xue, I. Liu, and M. Zhang, "Gaussian based particle swarm optimisation and statistical clustering for feature selection," in *Evolutionary Computation in Combinatorial Optimisation (EvoCOP)*, vol. 8600. Heidelberg, Germany: Springer, 2014, pp. 133–144.
[25] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, vol. 5. Orlando, FL, USA, 1997, pp. 4104–4108.
[26] B. H. Nguyen, B. Xue, and P. Andreae, "A novel binary particle swarm optimization algorithm and its applications on knapsack and feature selection problems," in *Proc. 20th Asia–Pac. Symp. Intell. Evol. Syst. (IES)*, Canberra, ACT, Australia, 2017, pp. 319–332.
[27] W.-N. Chen *et al.*, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Trans. Evol. Comput.*, vol. 14, no. 2, pp. 278–300, Apr. 2010.
[28] T. Hino, S. Ito, T. Liu, and M. Maeda, "Set-based particle swarm optimization with status memory for knapsack problem," *Artif. Life Robot.*, vol. 21, no. 1, pp. 98–105, 2016.
[29] J. Langeveld and A. P. Engelbrecht, "Set-based particle swarm optimization applied to the multidimensional knapsack problem," *Swarm Intell.*, vol. 6, no. 4, pp. 297–342, Dec. 2012.
[30] W. H. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, vol. 1. Cambridge, U.K.: Cambridge Univ. Press, 1988, p. 3.
[31] Q. Song, J. Ni, and G. Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 1–14, Jan. 2013.
[32] H. Al-Sahaf, A. Al-Sahaf, B. Xue, M. Johnston, and M. Zhang, "Automatically evolving rotation-invariant texture image descriptors by genetic programming," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 83–101, Feb. 2017.
[33] G. Patterson and M. Zhang, "Fitness functions in genetic programming for classification with unbalanced data," in *Proc. 20th Aust. Joint Conf. Artif. Intell. (AI)*, 2007, pp. 769–775.
[34] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Proc. Int. Conf. Database Theory*, London, U.K., 2001, pp. 420–434.
[35] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proc. 7th Int. Conf. Mach. Learn.*, 2000, pp. 359–366.
[36] M. Gutlein, E. Frank, M. Hall, and A. Karwath, "Large-scale attribute selection using wrappers," in *Proc. IEEE Symp. Comput. Intell. Data Min.*, 2009, pp. 332–339.
[37] M. Hall *et al.*, "The WEKA data mining software: An update," *ACM SIGKDD Explorat. Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.
[38] G. Azevedo, G. Cavalcanti, and E. Filho, "An approach to feature selection for keystroke dynamics systems based on PSO and feature weighting," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2007, pp. 3577–3584.
[39] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.

Binh Tran (S'14–M'18) received the B.E. degree in computer science from Can Tho University, Can Tho, Vietnam, in 1998, the M.Sc. degree in applied computer science from the Free University of Brussels, Brussels, Belgium, in 2002, and the Ph.D. degree in computer science from the Victoria University of Wellington, Wellington, New Zealand, in 2018.

She is currently a Post-Doctoral Research Fellow with the School of Engineering and Computer Science, Victoria University of Wellington. Her current research interests include evolutionary computation, feature manipulation including feature selection and construction, high-dimensional data, and machine learning.

Ms. Tran has been serving as a Reviewer for over ten international journals and conferences in the field, such as the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, *Applied Soft Computing*, IEEE CEC, GECCO, SEAL, and AAAI. She is a member of the IEEE Computational Intelligence Society.

Bing Xue (M'10) received the B.Sc. degree from the Henan University of Economics and Law, Zhengzhou, China, in 2007, the M.Sc. degree in management from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree in computer science from the Victoria University of Wellington, Wellington, New Zealand, in 2014.

She is currently a Senior Lecturer with the School of Engineering and Computer Science, Victoria University of Wellington. She has over 100 papers published in fully refereed international journals and conferences and most of them are on evolutionary feature selection and construction. Her current research interests include evolutionary computation, feature selection, feature construction, multiobjective optimization, image analysis, transfer learning, data mining, and machine learning.

Dr. Xue is currently the Chair of the IEEE Task Force on Evolutionary Feature Selection and Construction, IEEE Computational Intelligence Society (CIS), the Vice-Chair of the IEEE CIS Data Mining and Big Data Analytics Technical Committee, and the Vice-Chair of IEEE CIS Task Force on Transfer Learning and Transfer Optimization. She is also an Associate Editor/member of Editorial Board for five international journals and a Reviewer of over 50 international journals. She is the Finance Chair of IEEE Congress on Evolutionary Computation in 2019, the Program Co-Chair of the 31st Australasian AI in 2018, ACALCI 2018, and the 7th International Conference on SoCPaR2015, and she is also a tutorial chair, a special session chair, or a publicity chair for many other international conferences.

Mengjie Zhang (M'04–SM'10) received the B.E. and M.E. degrees from the Artificial Intelligence Research Center, Agricultural University of Hebei, Hebei, China, in 1989 and 1992, respectively, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 2000.

He is currently a Professor of computer science, the Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) with the Faculty of Engineering, Victoria University of Wellington, Wellington, New Zealand. He has published over 350 research papers in refereed international journals and conferences. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multiobjective optimization, feature selection and reduction, job shop scheduling, and transfer learning.

Prof. Zhang is currently chairing the IEEE CIS Intelligent Systems and Applications Technical Committee, and the immediate Past Chair for the IEEE CIS Emergent Technologies Technical Committee and the Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is the Vice-Chair of the IEEE CIS Task Force on Evolutionary Feature Selection and Construction and the Task Force on Evolutionary Computer Vision and Image Processing, and the Founding Chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a Committee Member of the IEEE NZ Central Section. He is a fellow of the Royal Society of New Zealand and have been a Panel member of the Marsden Fund (New Zealand Government Funding). He is also a member of ACM.